USING PDAL TO STREAMLINE LIDAR DATA PRODUCTS

Services Workflows & Processing

TNRIS Forum

October 25th 2023

Brent Porter

SETTING THE STAGE

LiDAR (Dr Jekyll)

- Undisputedly awesome source of elevation data for visualization & computation/modeling
- Currently we have statewide coverage in Texas thanks to the efforts of many groups and data champions
- Available for download and updated often by the good folks at ~~TNRIS~~ ahem TxGIO (I kid I kid)

SETTING THE STAGE

I do like the new name, really!

But I am also sad about that name!

Ok I am done. I promise*

SETTING THE STAGE

LiDAR (Mr Hyde)

- Laz & Las files are still a heavy lift for some folks
  - Particularly if you have a larger area or lots of areas you need to work with
  - Lots of space required for even regional coverage of LiDAR data
  - At UT we are hosting roughly 30TB of laz files right now (that number will only get bigger)
- Not only is it big 'on disk' – it takes lots of processing and therefore lots of time to work with the data.

What this means is that it just 'makes sense' for us to use the preprocessed DEMs which are in comparison smaller and preprocessed.

SETTING THE STAGE

DEMs
- DEMs **really** are a great compromise

But they ***are*** still a compromise
- The footprints & areas are predefined
- The filters on the point signals are also predefined

You get what I am saying. Nothing wrong with them but what if I want or need something different?

SETTING THE STAGE

So I started thinking?
Is there anything out there that could make working with the source point cloud data easier to build workflows and hopefully automated workflows for custom areas and custom output products?

THE CHALLENGE

Complexities

- Scaling a dense & complicated dataset for use on-demand
- Experimenting with performance of pdal processing and other spatial workflows
- Architecture for workflows and optimized point cloud queries

THE CHALLENGE

Is there a good solution for what I'd like to do?

THE CONTENDER

After some searching, I settled on

THE CONTENDER

PDAL!

PDAL is **P**oint **D**ata **A**bstraction **L**ibrary

- It is built conceptually like GDAL
- Command line tool with lots of features

THE CONTENDER PDAL

PDAL - Point Data Abstraction Library

PDAL is a C++ library for translating and manipulating point cloud data. It is very much like the GDAL library which handles raster and vector data. The About page provides high level overview of the library and its philosophy. Visit Readers and Writers to list data formats it supports, and see Filters for filtering operations that you can apply with PDAL.

In addition to the library code, PDAL provides a suite of command-line applications that users can conveniently use to process, filter, translate, and query point cloud data. Applications provides more information on that topic.

Finally, PDAL speaks Python by both embedding and extending it. Visit Python to find out how you can use PDAL with Python to process point cloud data.

The entire website is available as a single PDF at http://pdal.io/PDAL.pdf

THE CONTENDER
PDAL

Capabilities

- Works with las and laz files directly
- Lots of Readers, Writers & Filters
- Primarily Python – it is easy to integrate with gdal library tools (and other python libraries for complex workflow creation
- API connections to C++, Python as well as Java bindings (yum) & a DSL for Scala (a little more on this later)

THE CONTENDER PDAL

Ok but what does that mean to us?
Because that was some *nerdish* that just got thrown down!

I say that as an unrepentant nerd myself

THE CONTENDER
PDAL

To work with PDAL on the command line you invoke pdal while using conda environment mgmt. system to get all your libraries configured correctly – so → pdal pipeline <name/location of pipeline json file>

```
(base) C:\Users\Administrator>e:

(base) E:\>cd geodata\lidar_tiledb

(base) E:\Geodata\Lidar_TileDB>pdal pipeline pipeline-refilter-desert-mountain-alltiff-west-fileonly-defaultsmrf.json
```

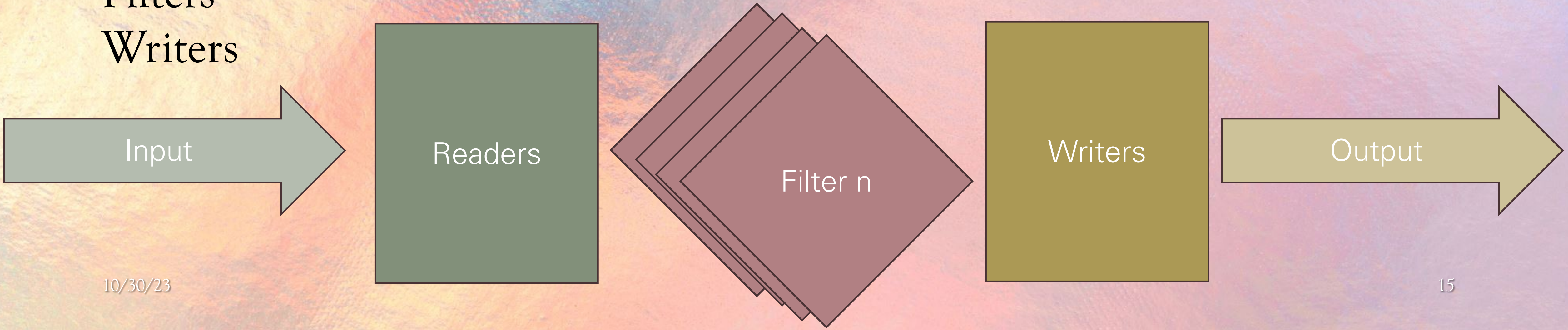
THE CONTENDER PDAL

The workflow for working with pdal includes 3 categories/components that you connect in a .json file to make that pipeline to generate output data (in many formats)

Readers

Filters

Writers



THE CONTENDER
PDAL

The quick takeaway on capabilities is that there are lots of pathways to using the library

AND

Reading between the lines - when a library feels confident enough to provide many pathways its because they have seen the value of the library and what it adopted as broadly as possible

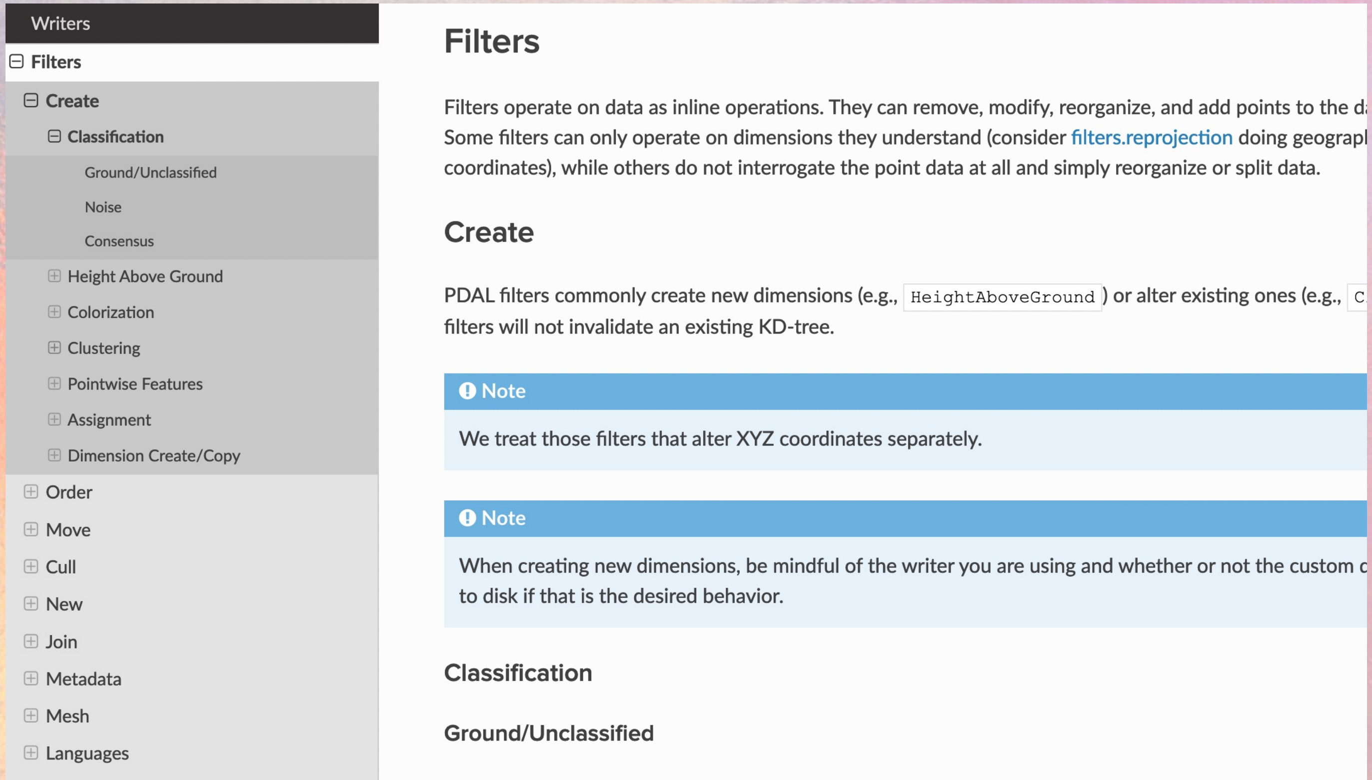
Also, all of those choices means it will be easier to onboard someone for PDAL work you will be more likely to find lots of interest and collaboration opportunities

THE CONTENDER PDAL

SOUND GOOD… Or at least possibly interesting?

LET'S LOOK AT FILTERS

- There are several filters that come with PDAL organized into categories
- These are ways of processing the lidar data in certain ways (including filtering the points to produce both DTMs and DSMs



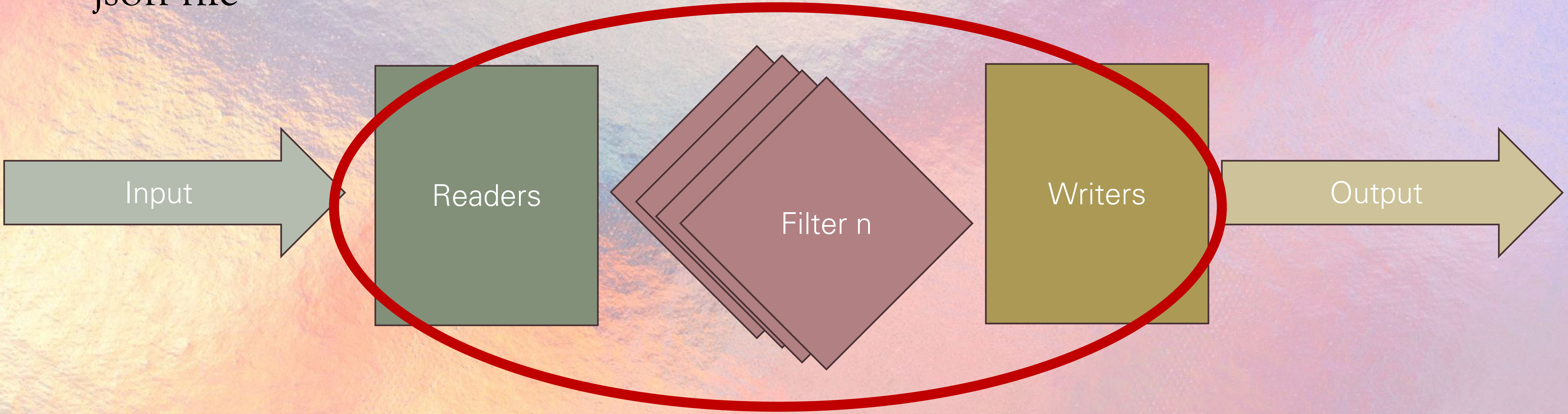
FILTERS

More Details

- These filters are also chainable!*
- You *could* for example, gather all of the building 'signals', filter it from a set of laz files for a particular extent or boundary, reproject them and then create a point csv that could then be used to provide elevation to 2d vector dataset you extracted with ML!
- There are over 70 filters all told categorized into ten categories.

LET'S REVIEW THE PIPELINE

Everything circled is part of the pipeline
json file



PIPELINES

Early Example: Chaining

pdal pipeline – querying data from point cloud db and filtering for bare-earth (DTM)

```
{
  "pipeline":[
    {
        "type":"readers.pgpointcloud",
        "connection":"host='localhost' dbname='geopointsdb' user='postgres' password='xxxxxxx' p
        "table":"sthsm",
        "column":"pa",
        "spatialref[illegible]PSG:26914"
    },
    {
      "type":"filters.assign",
      "assignment":"Classification[:]=0"
    },
    {
      "type":"filters.elm"
    },
    {
      "type":"filters.outlier"
    },
    {
      "type":"filters.smrf",
      "ignore":"Classification[7:7]",
      "slope":0.2,
      "window":16,
      "threshold":0.45,
      "scalar":1.2
    },
    {
      "type":"filters.range",
      "limits":"Classification[2:2]"
```

PIPELINES

Early Examples – wildcard for files

Most of these were small sets of 1, 2 or 4 laz files and primarily just for getting used to the process of building a set of filters and parameters for input/output

```
{
  "pipeline": [
    "ground/*.laz",
    {
      "type":"filters.range",
      "limits":"Classification[2:2]"
    },
    {
      "filename":"dtm-redux-3097412c.tif",
      "gdaldriver":"GTiff",
      "output_type":"all",
      "resolution":"1.0",
      "type": "writers.gdal"
    }
  ]
}
```

TERRAIN PRODUCTS

Early Example Result

DSM from LiDAR source for Port Arthur showing an industrial plant & surrounding neighborhoods

TERRAIN PRODUCTS

Early Example Result "Tx Sized"

TERRAIN PRODUCTS

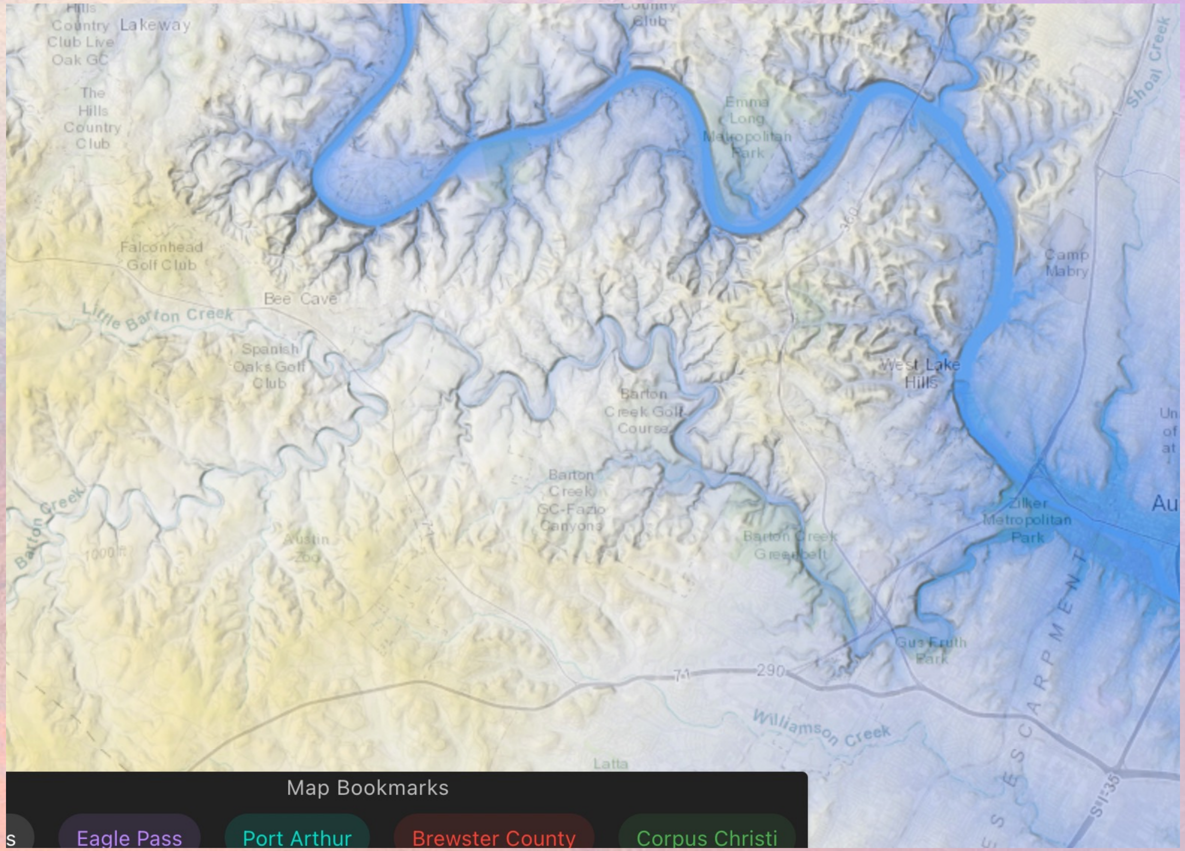
Early Examples

DSM from LiDAR source for Eagle Pass showing buildings and topography

TERRAIN PRODUCTS

Early Examples

Travis County hillshade & DTM with a custom color gradient



THESE SEEM VIABLE, INTERESTING AND FLEXIBLE AT FIRST GLANCE, RIGHT?

Well then…

Once more unto the breech Dear Cartographers, once more!

SO THEN LET'S TALK ABOUT….

METRICS!!!!!

WAIT! METRICS????

I know I know

But sometimes you just got to go there – last year I was discussing metadata and now metrics.

Its like I've turned into my father's geographer

WHAT IS METRICS REALLY WITHOUT AN EXPERIMENTAL ENVIRONMENT?

Hardware

Windows 2016 Server Virtual Machine

4 cores Dell "Gold" 6138 Xeon 2GHz

32 GB Ram

Software

PDAL 2.5.3 (Conda3/Python 3.10.10)

ArcGIS Pro 3.1

QGIS 3.4

METRICS

Lets look at processing laz files for eagle pass into a DSM
(for tiles – equate 1 tile with one laz file)
I am using a morphological filter and playing with the return numbers filtered as well as which bands for the output geotif

Single Tile	SMRF 2:2 to 7:7 Classification Min Tiff	3:48
8 Tiles	SMRF 2:2 to 6:7 Classification Mean Tiff	54:25:00
8 Tiles	2:2 Only Classification All Tiff	18:14
16 Tiles	2:2 Only Classification All Tiff	52:43:00
12 Tiles	2:2 Only Classification All Tiff	13:27

Classification Value and Meaning
0 Created, never classified
1 Unclassified
2 Ground
3 Low Vegetation
4 Medium Vegetation
5 High Vegetation
6 Building
7 Low Point (noise)
8 Model Key-point (mass point)
9 Water
10 Reserved for ASPRS Definition
11 Reserved for ASPRS Definition
12 Overlap Points
13-31 Reserved for ASPRS Definition

METRICS

Let's look -

Some oddities & chin scratchers…

Single Tile	SMRF 2:2 to 7:7 Classification Min Tiff	3:48
8 Tiles	SMRF 2:2 to 6:7 Classification Mean Tiff	54:25:00
8 Tiles	2:2 Only Classification All Tiff	18:14
16 Tiles	2:2 Only Classification All Tiff	52:43:00
12 Tiles	2:2 Only Classification All Tiff	13:27

Look at those times! Why the heck does it take less time for 12 laz files than 8????

METRICS

Looking deeper into provides some insight. Turns out that differences in laz file sizes can also affect the performance!

Single Tile	SMRF 2:2 to 7:7 Classification Min Tiff	3:48	
8 Tiles	SMRF 2:2 to 6:7 Classification Mean Tiff	54:25:00	
8 Tiles	2:2 Only Classification All Tiff	18:14	544 mb laz
16 Tiles	2:2 Only Classification All Tiff	52:43:00	1025 mb laz
12 Tiles	2:2 Only Classification All Tiff	13:27	329 mb laz

METRICS

Also using all bands in creating the tiff and then post-processing for the single band mean was WAY more efficient that doing it with PDAL

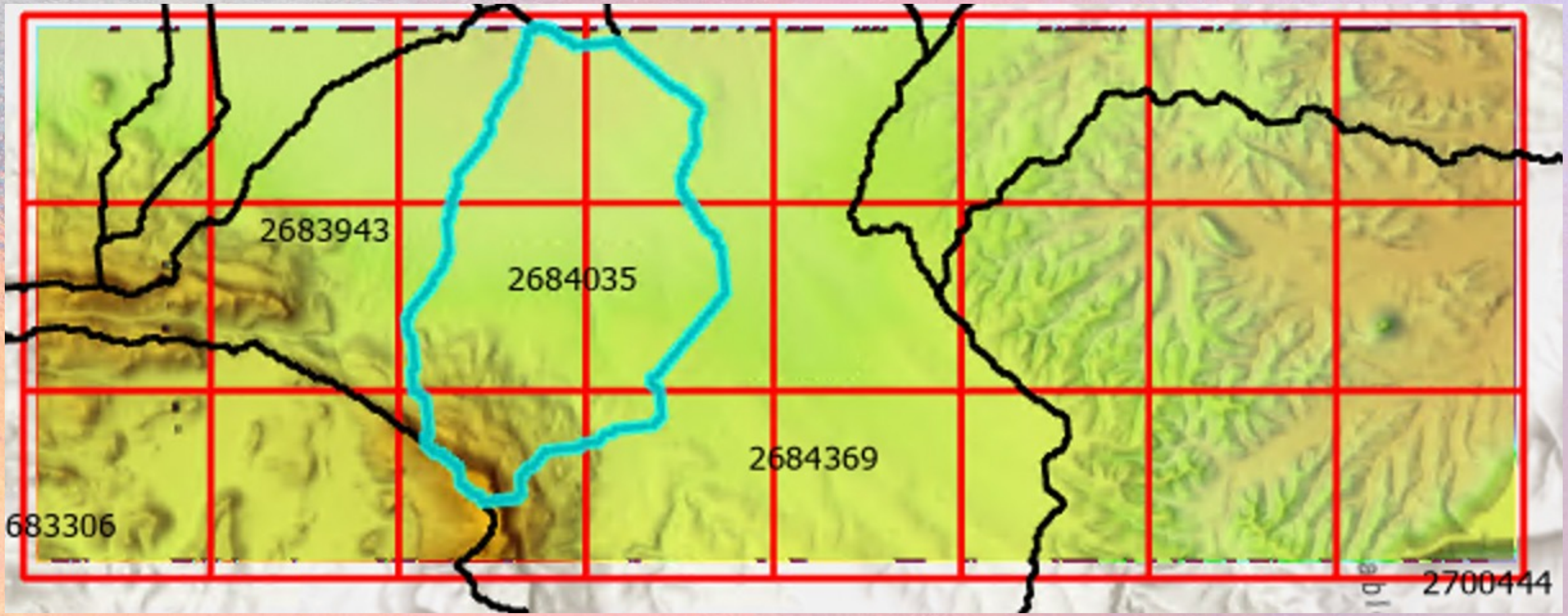
Single Tile	SMRF 2:2 to 7:7 Classification Min Tiff		3:48	
8 Tiles	SMRF 2:2 to 6:7 Classification Mean Tiff		54:25:00	
8 Tiles	2:2 Only Classification All Tiff		18:14	544 mb laz
16 Tiles	2:2 Only Classification All Tiff		52:43:00	1025 mb laz
12 Tiles	2:2 Only Classification All Tiff		13:27	329 mb laz

METRICS – GIVE ME SOME PERSPECTIVE HERE!

How big an area is this
12 laz files is generally 2 or more huc 8s but your mileage may vary depending on the area

This is the area in W Tx I used for this talk – USGS 2019 Desert Mt

24 laz files & 70cm data

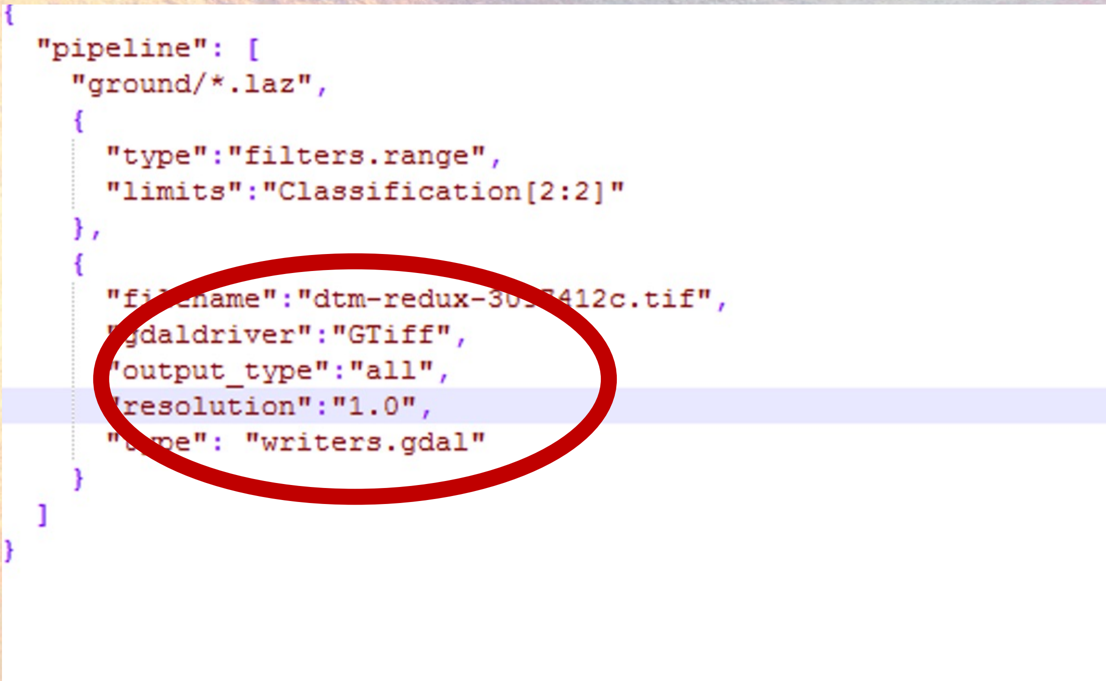


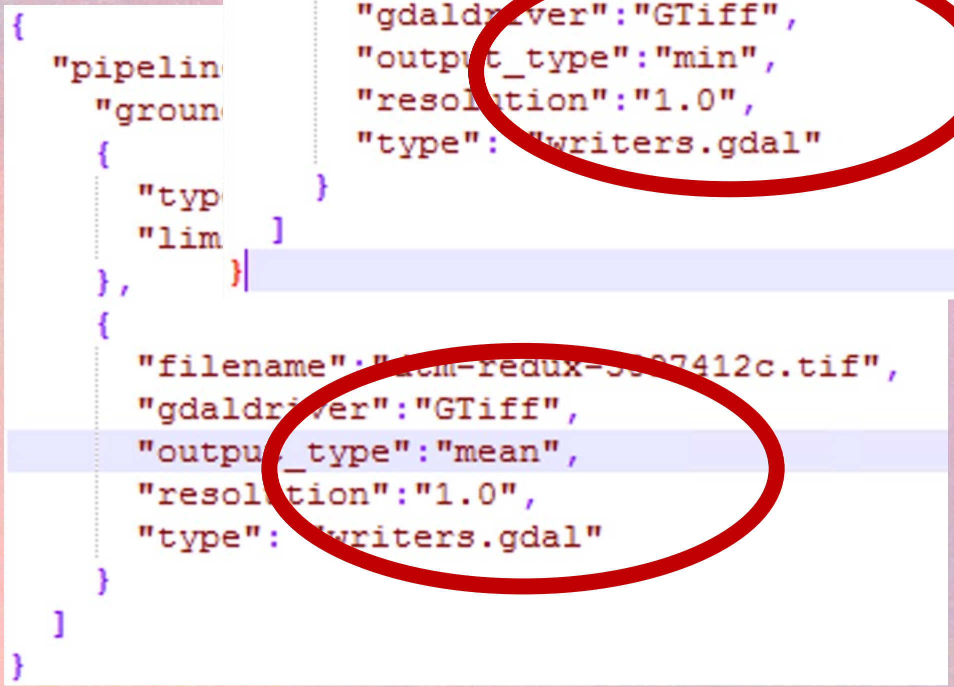
METRICS

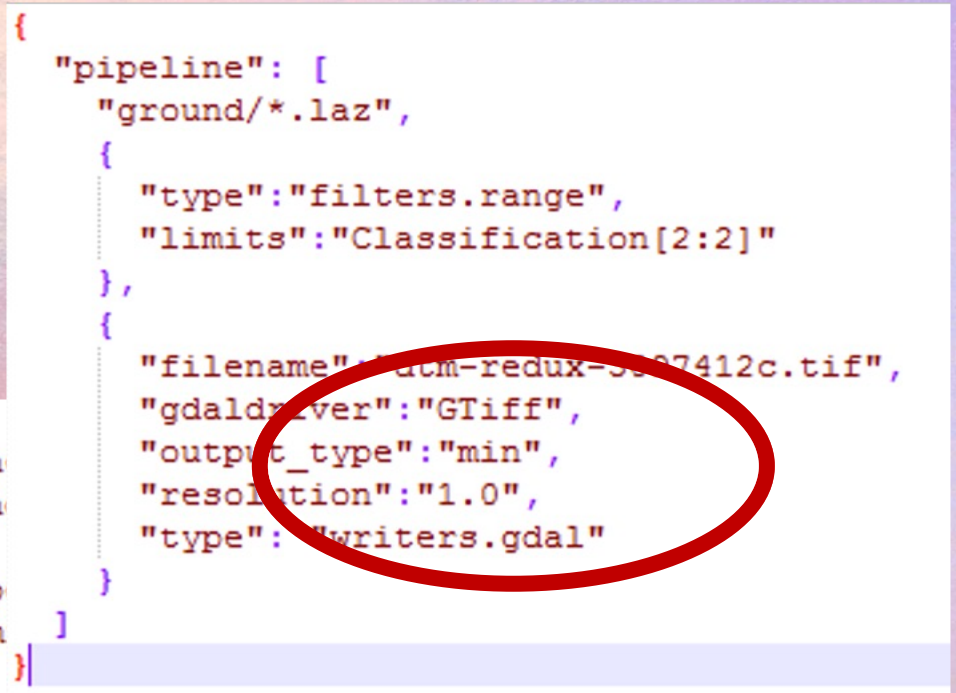
Back to output_type in the writer

Here are some shots of different choices

When you choose 'all' you get 'bands' for min, max, mean, idw, count & stdev







METRICS

Winner Winner Chicken Dinner!!

From those earlier metrics we can see that it is 3 times faster (on average) to use "all" rather than using the more selective options

```
{
  "pipeline": [
    "ground/*.laz",
    {
      "type":"filters.range",
      "limits":"Classification[2:2]"
    },
    {
      "filen[illegible]":"dtm-redu[illegible]097412c.tif",
      "gdaldriver":"GTiff",
      "output_type":"all",
      "resolution":"1.0",
      "type": "writers.gdal"
    }
  ]
}
```

METRICS

To summarize those findings

The processing performance and time to completion are dependent on

1. *Number & type of Filter (remembering also that filters can be chained!)*
2. *Attributes of those Filters (for example – the difference in time for generating tifs with all 'bands' vs selecting min/max/mean)*
3. *Characteristics of your point cloud data (overall size, density of points, resolution, etc.)*

METRICS – ALSO!

Database Source (pg-pointcloud/postgres) vs File-based

PG-PointCloud/PostgreSQL can store point cloud data that can be accessed via a reader in pdal

- *Initial cost does take time – 24 laz files took 54:14 to load*
  - *That's non-trivial BUT the pay off is worth it in my testing!*

First time is the File-based & the second is for the DB Call

	Default SMRF File-based				Default SMRF Filter Postgres		
45:13:00	~ 2 Tiles		East of 12	13:15	12 Tiles		East of 12
45:17:00	~ 2 Tiles		West of 12	13:17	12 Tiles		West of 12

METRICS

DB vs File-based continued

Database sources are consistently 3 times faster using the same filters and the same laz files to generate DTMs

```
{
 "pipeline":[
   {
      "type":"readers.pgpointcloud",
      "connection":"host='localhost' dbname='geopointsdb' user='postgres' password='xxxxxxx' port='5432'",
      "table":"jeff",
      "column":"pa",
      "spatialreference":"EPSG:26913",
      "where":"PC_Intersects(pa, ST_MakeEnvelope(482618, 3497387, 488396, 3493134, 26913))"
   },
   {
      "type":"filters.smrf"
   },
   {
    "type":"filters.range",
    "limits":"Classification[2:2]"
   },
   {
      "type":"writers.gdal",
      "filename":"desert-mountain-usgs2019-where-defaultsmrf.tif",
      "output_type":"all",
      "gdaldriver":"GTiff",
      "window_size":3,
      "resolution":1.0
   }
   ]
}
```

```
{
 "pipeline":[
   {
  "type":"readers.las",
     "filename":"desert_mountain_usgs19_70cm3/file-test/*.laz",
     "spatialreference":"EPSG:26913"
   },
   {
      "type":"filters.smrf"
   },
   {
     "type":"filters.range",
     "limits":"Classification[2:2]"
   },
   {
      "type":"writers.gdal",
      "filename":"desert-mountain-usgs2019-where-fileonly-defaultsmrf.tif",
      "output_type":"all",
      "gdaldriver":"GTiff",
      "window_size":3,
      "resolution":1.0
   }
   ]
}
```

METRICS

DB vs File-based continued

*And **remember** - that is doing an arbitrary query (meaning you can build any where clause to query your point cloud table) against the database for an extent using PostgreSQL's PostGIS spatial extension*

METRICS

DB vs File-based continued – sample where clause in UTM Z13

```
{
  "pipeline":[
    {
        "type":"readers.pgpointcloud",
        "connection":"host='localhost' dbname='geopointsdb' user='postgres' password='xxxxxxx' port='5432'",
        "table":"jeff",
        "column":"pa",
        "spatialreference":"EPSG:26913",
        "where":"PC_Intersects(pa, ST_MakeEnvelope(482618, 3497387, 488396, 3493134, 26913))"
    },
    {
        "type":"filters.smrf"
    },
    {
      "type":"filters.range",
      "limits":"Classification[2:2]"
    },
    {
        "type":"writers.gdal",
        "filename":"desert-mountain-usgs2019-where-defaultsmrf.tif",
        "output_type":"all",
        "gdaldriver":"GTiff",
        "window_size":3,
        "resolution":1.0
    }
    ]
}
```

WHAT IS NEXT??

There are lots more things to work on…

- Running the PDAL processes on Lonestar 6 – one of UTs HPC environments over at TACC – by threading out the process to many nodes can we see orders of magnitude increases in performance?
- Increasing the memory allocation for the hardware running PDAL

BUT WAIT THERE'S MORE!!

FUTURE COOL

PDAL also offers Java Bindings! & sample code to work with Scala (a jvm-compiled language built specifically for parallel processes and threading) – running tests with Scala & Java could also enhance performance as generally java programs run quicker than python-based scripts

IS IT POSSIBLE TO OPERATIONALIZE THIS WORKFLOW?

While building and running these workflows are interesting, is it possible that we could provide a service that allows for custom download of point cloud data for a custom area?

We will hopefully answer that as the experiments progress!

THANK YOU ALL FOR LISTENING TO ME RANT AND DRONE ON ABOUT THIS!

Questions?

Thanks also to TACC & all of the great folks over there that we are working with and my good teammates at CSR. And the folks at TNRIS (hee hee) for providing LiDAR for Texas*

* Ok **NOW** I am done